Rule WLM006: The response performance goal may be too large

Finding: CPExpert believes that the response performance goal specified for a service class may be too large.

Impact: This finding should be viewed a LOW IMPACT or MEDIUM IMPACT on the performance of your computer system. The finding could have a HIGH impact on the performance of the service class identified by this finding.

Logic flow: This a basic finding. There are no predecessor rules.

Discussion: Users specify a performance goal for each service class. There are four types of performance goals: average response, percentile response, execution velocity, and discretionary. The first two (average response and percentile response) are the subject of this rule description.

The Workload Manager ISPF Response Time Goal Panel allows a response performance goal of up to 24 hours to be specified. Response goals in minutes or hours are typically associated with batch workloads.

CPExpert believes that a response performance goal of over 5 minutes is likely to result in unsatisfactory performance in most environments and a response goal of **less than 1 minute** is more likely to yield desired results. The following discussion explains why CPExpert believes that relatively long response goals are inappropriate:

- The Workload Manager attempts to adjust system resources as necessary to achieve the performance goal specified for service classes. The Workload Manager evaluates how well the existing resource policy allows performance goals to be met **every 10 seconds**. This 10-second process is called the *Policy Adjustment Interval*.

  During policy adjustment, the Workload Manager evaluates the performance of each service class. The evaluation is accomplished by computing a Performance Index for each service class period and analyzing the Performance Index within each level of Goal Importance[1]

  Obviously, in order to analyze how well a service class is performing against a response goal, one or more "transactions" must have completed during the previous interval. If no transactions completed, the Workload Manager has no information on which to assess the

[1]Please refer to Section 4 for a more comprehensive discussion of the Policy Adjustment process.

performance of the service class with respect to the response goal. In fact, the process works much better if *many* transactions complete, as the Workload Manager can compute average response or percentile response based on a larger sample of work.

Once the Workload Manager makes a policy adjustment decision, it evaluates the effect of that decision during the **next** policy adjustment interval. In order to assess the effect of the decision on response time, multiple transactions must complete so the Workload Manager can evaluate the effect on transaction response[2] time.

In summary, the Workload Manager must detect that a response goal was missed, and take action to improve performance for the service class missing its goal. Then the WLM must determine whether the action helped, or whether additional actions must be taken. This cycle can continue for awhile. With short response goals and lots of transactions, the WLM will have adequate performance data (many ended transactions yielding response information) to evaluate, and will have quick feedback on how well its decisions helped the service class meet its responses goal. The WLM can detect/adjust/evaluate/adjust relatively frequently with respect to the goal. Still, the detect/adjust/evaluate happens only once per 10 seconds.

This process works extremely well if the transactions represent interactive work (e.g., TSO transactions, CICS transactions, or IMS transactions). Many transactions normally will complete in a policy adjustment interval and the Workload Manager will have adequate information upon which to assess the results of the policy decisions.

If the "transaction" really is a batch job with a relatively long response performance goal, it is unlikely that many transactions will complete in the 10-second policy adjustment interval. Thus, the Workload Manager has little or no information on which to base its policy adjustment decisions; the Workload Manager must wait for batch jobs to complete before any decisions can be made. Consequently, the Workload Manager will be unresponsive in adjusting system resources to meet the performance goal for the batch jobs.

- Consider that the WLM makes resource adjustment decisions every 10 seconds. These adjustments are partially based on how well work meets goals (other factors are general housekeeping, etc.).

[2]The Workload Manager can assess the effect of some policy decisions without response-related information. For example, suppose that the Workload Manager determined that paging was a major cause of performance degradation. The Workload Manager might make processor storage decisions to either protect or restrict central or expanded storage for certain service classes. The effect of these decisions would be apparent from a system view (e.g., paging increased or decreased) without requiring transaction response data. However, the Workload Manager cannot determine whether the overall response performance goal is being met until transactions complete.

Suppose that a few batch jobs execute in a service class period with a *response goal of 20 minutes*. It takes awhile[3] for the WLM to detect that a goal was missed. If, for example, a job took over an hour to complete, it would take more than an hour for the WLM to recognize that work in the service class period was missing its response goal.

Recall that the WLM is going to make decisions every 10 seconds, and the WLM then must determine whether the decisions helped improve response time. In this example, more than 20 minutes additional must lapse before the WLM can have data to figure out whether its decisions were appropriate! In fact, at least one transaction must end before the WLM can assess whether the performance goal had been achieved. If a new job would take an hour to complete (meaning that the policy adjustment decisions did not help), it might take the WLM that hour to determine that its policy adjustment was not effective.

It is true that the WLM can make resource allocation decisions based on observed delays to the long-running work (denied CPU use, paging delays, I/O delays, etc.), and the WLM can dynamically assess whether the work was being delayed less because of decisions related to these CPU delays, paging delays, I/O delays, etc. Consequently, the WLM can "guess" that performance is improving based on decreased delay to the work. However, that is exactly what execution velocity takes into account. This means that for long transaction response times, the WLM in effect implements velocity goal management.

This “implicit” implementing velocity goal management is not as effective as explicitly stating a velocity goal. This is because it takes too long (the duration of the response goal) for the WLM to detect that a response goal was missed, whereas execution velocity goals would be computed every 10 seconds.

- The Workload Manager evaluates system performance considering the performance of all service classes, based on their level of importance. Most modern computer environments have a mix of workload, consisting of both interactive and non-interactive. The interactive workload usually has a higher importance, and interactive workload often is quite dynamic in terms of system requirements.

  One consequence of this nature of interactive work is that the Workload Manager typically will adjust resource allocation policies based on the requirements of the interactive workload. Only in the most stable environments will policy adjustment decisions be driven by relatively lengthy non-interactive response goals.

[3] That time would be more than 20 minutes, since at least one batch job must complete and exceed its 20 minutes goal before the WLM could detect that the goal had been missed.

- Workload Manager developers have stated that only 20 minutes of historical information are retained by the Workload Manager. At present, it is unclear how the internal Workload Manager algorithms discard data and it is unclear what effect discarding data has on lengthy response goals.

  Very short transactions are typically homogeneous with respect to their execution characteristics so the WLM does not have to worry about radically differing use of processor or I/O amongst the different transactions. Even if some transactions use radically differing resources from the general population, their effect will be minuscule because they end so quickly. Consequently, an adjustment decision can be made without worry that the resource demands will radically change from one "transaction" to the next.

  These characteristics of short transactions do not normally apply with batch work or other work that consists of long-running transactions. Any particular long-running batch job is not necessarily homogenous with the batch job population, with respect to its use of system resources. Also, unlike short transactions, long-running work does not tend to be homogeneous, and there often is drastic differences in the resource requirements among long-running jobs.

  This heterogeneous nature of long-running work would often result in the WLM making policy adjustment decisions, based on resource consumption characteristics of **ended** long-running work. However, the **currently-running** work might not have similar resource demands and delays that the WLM had observed from the ended work.

- Please note that IBM's *MVS/ESA SP Version 5: Planning: Workload Management* specifically states "Work that is appropriate for a response goal should have a reasonable number of transaction completions over 20 minutes of time. If there are only a few completions, you are better off using a velocity goal."

- You might have defined service classes to describe subsystem transactions (such as CICS transactions) which have long Idle state times. Rule WLM122 describes transactions with long Idle state times, and suggests an approach which includes defining a **very long** response goal for the service class containing these transactions. CPExpert suppresses Rule WLM006 for transaction subsystem service classes.

The following example illustrates the output from Rule WLM006:

```
RULE WLM006:  THE RESPONSE PERFORMANCE GOAL MAY BE TOO LARGE

 CPExpert noticed that BATDEVHI Service Class (Period 1) had a response
 goal of 0:20:00.0.  This response goal is quite large relative to the
 intervals in which the Workload Manager makes system adjustments.  It
 is unlikely that the Workload Manager's actions will be effective with
 such a large response goal.  You may have better success by specifying
 an execution velocity goal for this service class.  Please refer to
 Rule WLM006 in the WLM Component User Manual for a discussion of this
 issue.  Rule WLM173 in this output will show any measurement intervals
 when CPExpert believed that the large response goal could be a cause
 of the service class missing its response goal.  If Rule WLM173 is not
 produced, the large response goal probably had no adverse effect on
 this service class period.
```

Suggestion: CPExpert suggests that you consider the following alternatives if Rule WLM006 is produced:

- Specify an execution velocity goal for the service class identified by Rule WLM006. This normally is the best alternative. There are exceptions to this general advice, as discussed below.

- Specifying ANY goal automatically means that work elements in the service class will be assigned to the range of dispatching priorities reserved for "goal" work. This means that the work will always have a higher dispatching priority than discretionary work. Consequently, specifying a long response goal could be a valid approach if you want to always make sure that the work has a higher dispatching priority than discretionary. As described earlier, however, this is not normally a good solution since specifying an execution velocity goal (even a small velocity) would provide better WLM actions.

- Specifying a long response goal causes the work to be a candidate for Discretionary Management. While the work will always have a higher dispatching *priority* than discretionary, a very high goal **could** cause it to have a Performance Index less than 0.7 (which is the Performance Index when cap slice capping can start), and stops internal resource capping when the Performance Index is greater than or equal to 0.81 (which is the internally-used Performance Index for discretionary work). Consequently, the work with a long response goal can be subject to the Discretionary Management Cap Slice algorithm, to allow discretionary to periodically have access to the CPU. Again, this is not normally a good solution since specifying an execution velocity goal (even a small velocity) would provide better WLM actions.

- **Rule WLM173** will show any measurement intervals when CPExpert believed that the large response goal could be a cause of the service class missing its response goal. If Rule WLM173 is not produced, the

large response goal probably had no adverse effect on this service class period.

- You can adjust (or turn off) this analysis if you disagree with CPExpert's reasoning. The **MAXRESP** guidance variable in USOURCE(WLMGUIDE) can be used to provide guidance to CPExpert on the maximum response performance goal which CPExpert views as acceptable.

  The default specification for the MAXRESP guidance variable in WLMGUIDE is %LET MAXRESP=0:05:00, indicating that CPExpert that any response performance goal greater than 5 minutes causes Rule WLM006 to be produced. You could "turn off" this rule by specifying %LET MAXRESP=24:00:00 in USOURCE(GENGUIDE). Since a response goal cannot be larger than 24 hours, this would have the effect of "turning off' CPExpert's analysis in this area.

Reference: MVS Planning: Workload Management

MVS/ESA(SP 5):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V1R1):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V1R2):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V1R3):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R4):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R5):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R6):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R7):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R8):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R9):	Chapter 8: Defining Service Classes and Performance Goals
OS/390 (V2R10):	Chapter 8: Defining Service Classes and Performance Goals
z/OS (V1R1):	Chapter 8: Defining Service Classes and Performance Goals
z/OS (V1R2):	Chapter 8: Defining Service Classes and Performance Goals
z/OS (V1R3):	Chapter 8: Defining Service Classes and Performance Goals
z/OS (V1R4):	Chapter 8: Defining Service Classes and Performance Goals

"Migrating to the MVS Workload Manager", Peter Enrico (IBM Corporation Workload Manager developer), 1995 SHARE Winter Meeting

